

## 作業三—Hopfield 書面報告

### 1. 程式簡介

將程式分成訓練、回想、GUI 介面

(a) **Hopfield\_training.py(訓練)**：設定一個物件名為 Hopfield\_Train，在新增此物件時，需傳入訓練資料集及此訓練資料集的矩陣維度。為了方便說明，設  $N$  為訓練資料矩陣的列數乘行數的值。在此物件中有兩個函式。第一個函式為 `compute_weight`，用來計算 Hopfield 的鍵結值。在此函式中會先初始化鍵結值為一方塊矩陣，行數和列數的值都是  $N$ 。除了初始化鍵結值外，也會設定一個大小為  $N$  乘  $1$  的  $X$  向量記錄訓練資料。在訓練時，迭代訓練資料的每一行，在每一次的迭代中，若資料為  $1$ ，則設  $X$  向量的相對應位置為  $1$ ，若資料為空白，則設  $X$  向量的相對應位置為  $-1$ 。當迭代完其中一筆訓練資料後，將  $X$  乘以  $X$  的轉置向量，並將結果加入鍵結值中。最後回傳鍵結值。

第二個函式名為 `regularize`。在 `regularize` 中，以  $W = \frac{1}{P}W - \frac{n}{P}I$  ( $P$  為  $N$ ， $n$  為訓練資料中訓練資料圖樣的個數) 的方式，正規化鍵結值。另外也在此函式中設定 Hopfield 的  $\theta$  為大小為  $N$  乘  $1$  的零向量。

(b) **Hopfield\_testing.py(回想)**：設定一個物件名為 Hopfield\_Test，在新增此物件時，需傳入測試資料集及訓練後的 Hopfield\_Train 物件。此物件中又分為 3 個函式。第一個函式為 `test_transform`，此函式的目的是將資料以空白和  $1$  的表達方式分別轉成  $-1$  和  $1$  來呈現，並將所有轉換後的圖樣記錄到一列表中。第二個函式為 `train`，此函式是用非同步的方式更新測試資料：迭代鍵結值中的每一列，將每一列乘上轉換後的測試資料，若相乘後的結果大於  $\theta$  值，則將測試資料的相對應列設為  $1$ ，反之若小於  $\theta$  值，將測試資料的相對應列設為  $-1$ ，而若等於  $\theta$  值，則不更新測試資料的數值。若更新後的測試資料和訓練資料一樣，或是更新次數達到設定的 `epoch` 值後，即停止迭代更新。第三個函式為 `print_test`，此函式是為了在 GUI 介面中呈現回想結果而設定的。在此函式中，將更新後的測試資料寫入名為“`print_after.txt`”的文字檔中。若測試資料為  $1$ ，則寫入  $1$ ，若資料不為  $1$ ，則寫入空白。當寫完一個測試圖樣後，也寫入一空白行分隔圖樣。

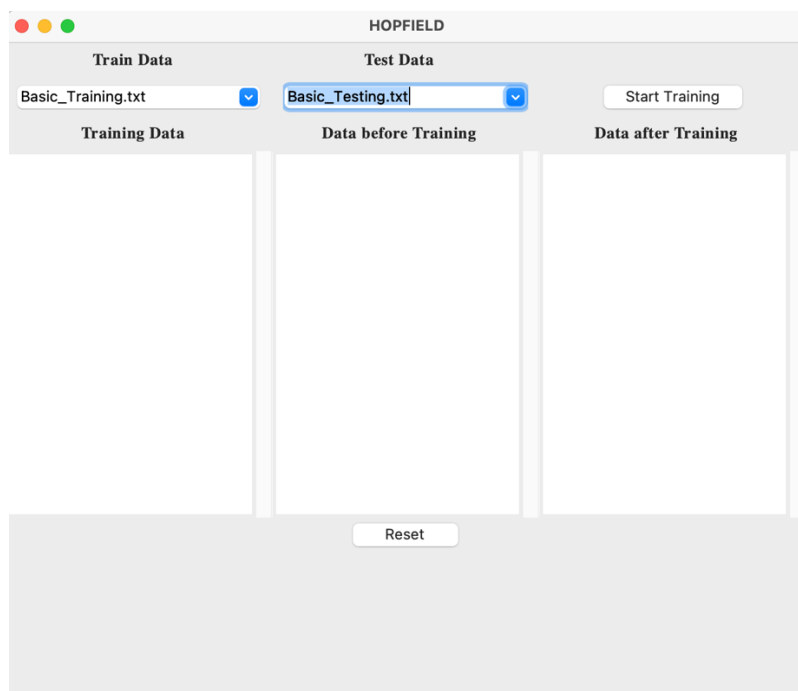
(c) **HopfieldTK.py(GUI 介面)**：此為控制 GUI 介面的檔案。在此 GUI 中會有兩個函式。第一個函式為 `RESET`，此函式為用來清空頁面中呈現圖像的文字欄。第二個函式為 `train`，此函式會抓取使用者選擇的訓練資料集和測試資料集。讀取訓練資料集和原始的測試資料集，寫入呈現兩資料集圖樣的文字欄中。再判斷使用者選擇的訓練資料集是“`Basic_Training.txt`”還是“`Bonus_Training.txt`”。若使用者選擇的是“`Basic_Training.txt`”，則新增一個訓練物件(Hopfield\_Train)，傳遞參數的列數及行數分別為  $12$  和  $9$ ；若使用者選擇的是“`Bonus_Training.txt`”，則也則新增一個訓練物件(Hopfield\_Train)，傳遞參數的列數及行數都為  $10$ 。除了訓練資料集外，也要判斷使用者選擇

哪一個資料集當作測試資料集。新增一個 `Hopfield_test` 物件，傳入測試資料集和對應的 `Hopfield_train` 後，開始進行回想。回想後，透過在 `Hopfield_testing` 產生的 `"print_after.txt"`，將回想後的結果寫入回想後的文字欄裡。

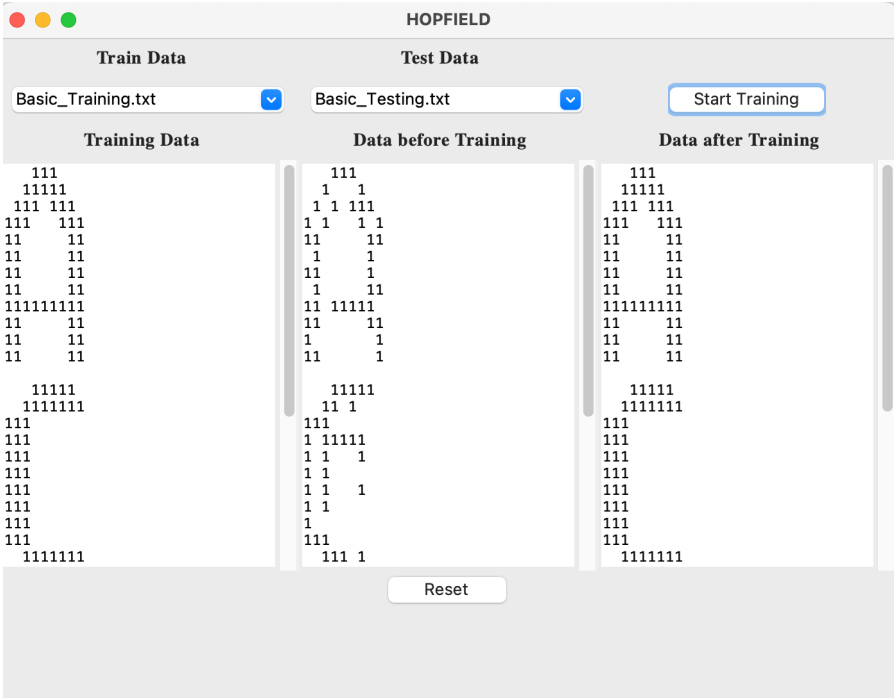
(d) **change.py(訓練資料加入雜訊(加分項目))**：此為將訓練資料加入雜訊的檔案。先讀取要加入雜訊的訓練資料，再利用一個迴圈取得訓練資料的維度。在迭代每一行的訓練資料中，隨機產生 1 或 0(1 代表加入雜訊，0 代表不加入雜訊)，產生 1 的機率為 0.25，產生 0 的機率為 0.75。若原始訓練資料為空白，隨機產生的數為 0，或原始資料為 1，隨機產生的數為 1，則將訓練資料轉換成 -1；反之若訓練資料為空白，隨機產生的數為 1，或訓練資料為 1，隨機產生的數為 0，則將訓練資料轉成 1。當加入雜訊到一個圖樣後，就將此圖樣加入一個列表中。最後全部圖樣都加入雜訊後，開始迭代儲存雜訊圖樣的列表，並以一個空字串開始，若雜訊圖樣的元素為 1，則將字串加入 1，反之則加入空白。當每迭代一個圖樣，就將此字串加入空行，分隔圖樣。最後若是要將 `"Basic_Training.txt"` 加入雜訊，則將此字串寫入一名為 `"Basic_Change_Testing.txt"` 檔；若是要將 `"Bonus_Training.txt"` 加入雜訊，則將此字串寫入一名為 `"Bonus_Change_Testing.txt"` 檔。

## 2. 程式執行說明

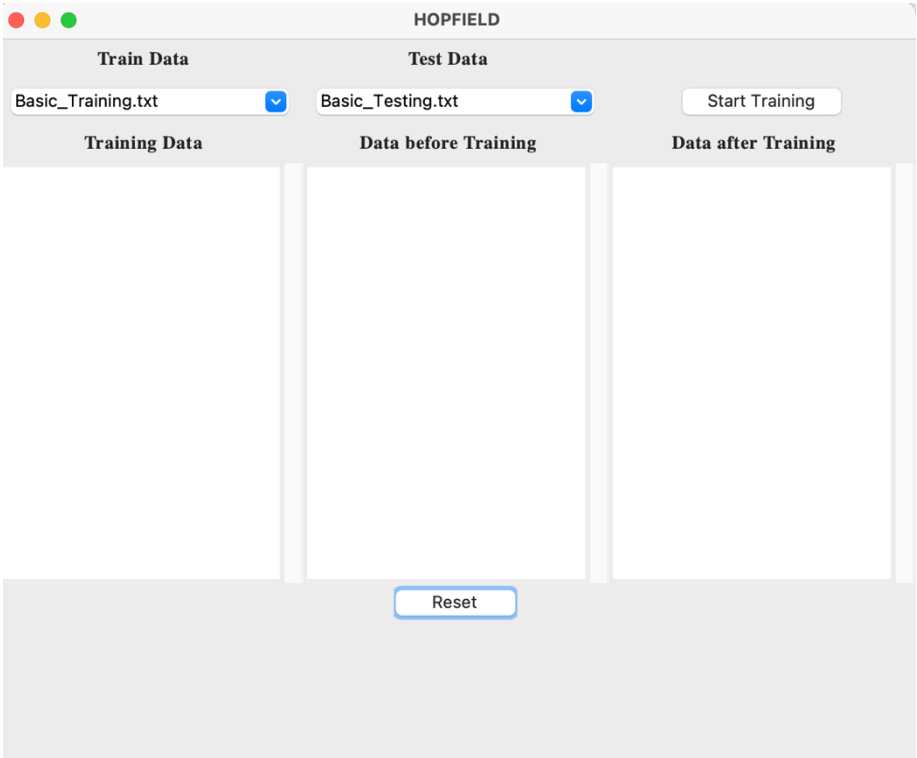
在 GUI 介面中，會有兩個 combo box，使用者可以選擇訓練資料集和測試資料集。若選擇 `"Basic_Training.txt"` 當作訓練資料集，使用者就只能選擇 `"Basic_Testing.txt"` 和已經先加入雜訊的訓練資料集 `"Basic_Change_Testing.txt"` 當作測試資料集；若選擇 `"Bonus_Training.txt"` 當作訓練資料集，使用者就只能選擇 `"Bonus_Testing.txt"` 和已經先加入雜訊的訓練資料集 `"Bonus_Change_Testing.txt"` 當作測試資料集。



按下”Start Training”按鈕後，就可以開始訓練，並進行回想。回想後，左邊的文字欄(Training Data)會印出訓練資料集的圖樣，中間的文字欄(Data before Training)會印出測試資料集回想前的圖樣，右邊的文字欄(Data after Training)會印出測試資料集回想後的圖樣。



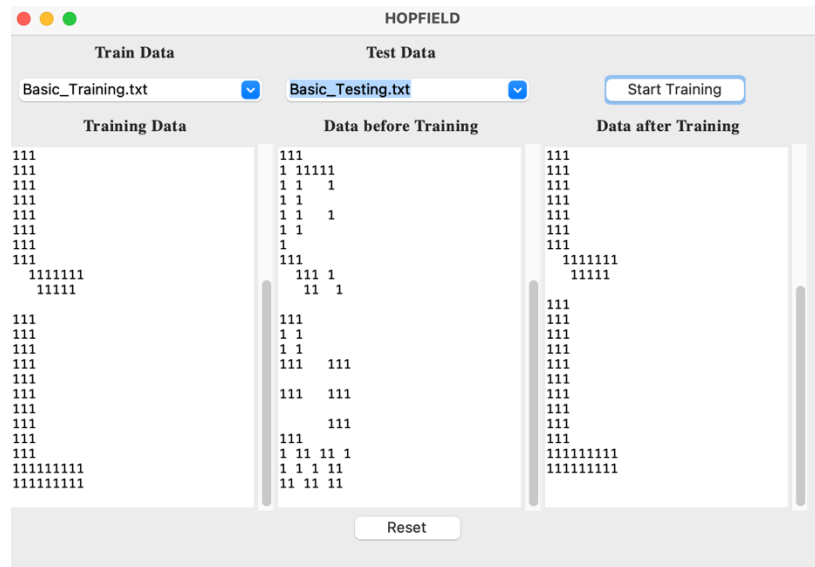
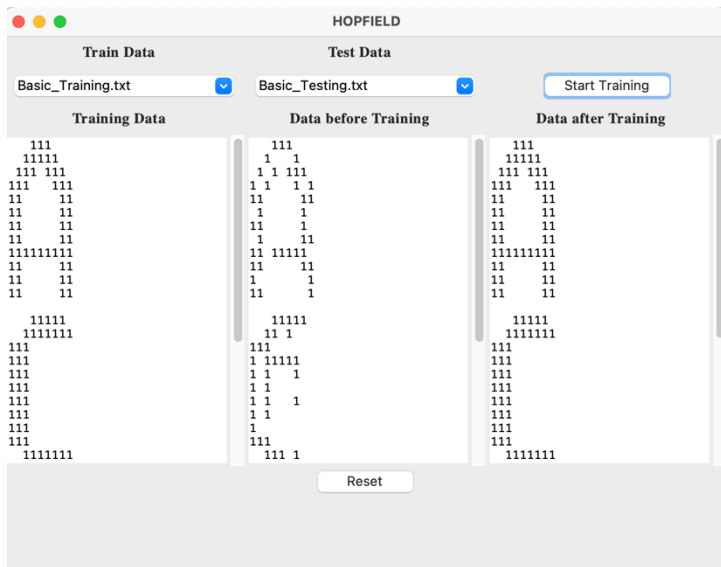
按下 Reset 按鈕後，會將三個文字欄清空，使用者就可以再重新選擇訓練資料集和測試資料集進行訓練及回想。



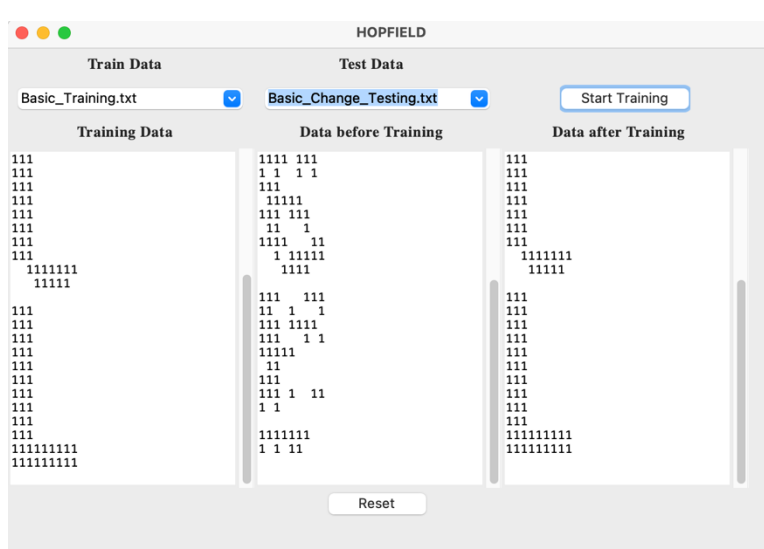
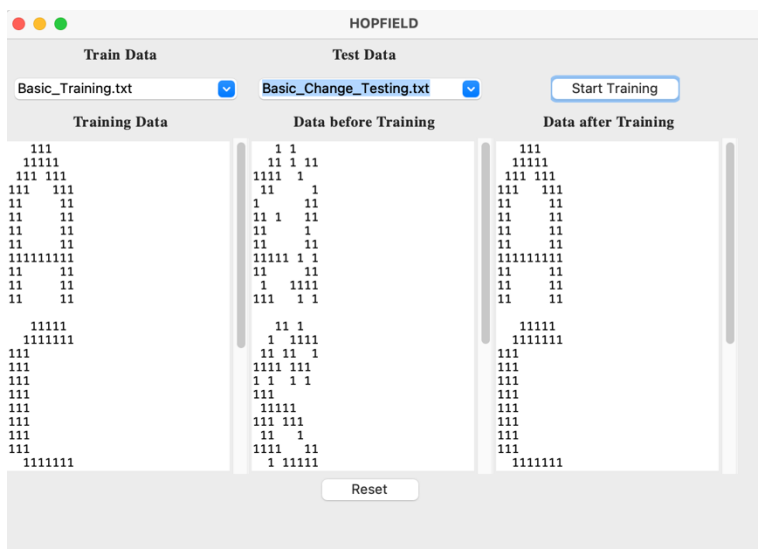
### 3. 實驗結果

基本題(Basic\_Testing.txt)、基本題的訓練資料加入雜訊當作測試資料(Basic\_Change\_Testing.txt)、加分題(Bonus\_Testing.txt)、加分題的訓練資料加入雜訊當作測試資料(Bonus\_Change\_Testing.txt)都可以回想成功：

(a) Basic\_Testing.txt



(b) Basic\_Change\_Testing.txt(加分)



The figure displays two instances of the HOPFIELD neural network simulation interface, showing the process of training and testing the network on binary data patterns.

**Left Window:**

- Train Data:** Contains 16 binary patterns (e.g., 1 1 1 1 1, 1 1 1 1 1, ..., 1 1 1 1 1).
- Test Data:** Contains 16 binary patterns (e.g., 1 1 1 1 1, 1 1 1 1 1, ..., 1 1 1 1 1).
- Training Data:** Shows the patterns used for training.
- Data before Training:** Shows the patterns used for testing before training.
- Data after Training:** Shows the patterns used for testing after training. The output patterns are mostly correct, with some minor errors (e.g., 1 1 1 1 1, 1 1 1 1 1).
- Buttons:** "Start Training" and "Reset".

**Right Window:**

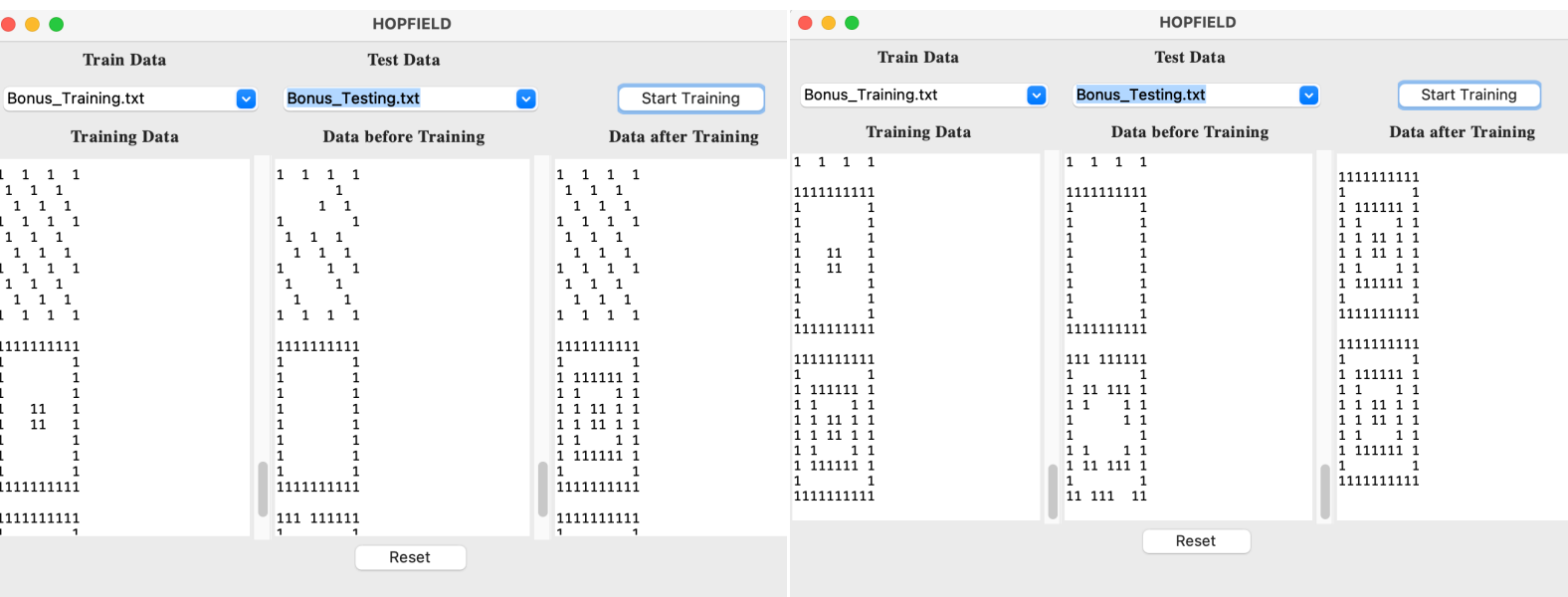
- Train Data:** Contains 16 binary patterns (e.g., 1 1 1 1 1, 1 1 1 1 1, ..., 1 1 1 1 1).
- Test Data:** Contains 16 binary patterns (e.g., 1 1 1 1 1, 1 1 1 1 1, ..., 1 1 1 1 1).
- Training Data:** Shows the patterns used for training.
- Data before Training:** Shows the patterns used for testing before training.
- Data after Training:** Shows the patterns used for testing after training. The output patterns are mostly correct, with some minor errors (e.g., 1 1 1 1 1, 1 1 1 1 1).
- Buttons:** "Start Training" and "Reset".

The screenshot shows the HOPFIELD application interface. At the top, there are three colored window control buttons (red, yellow, green) on the left and the title 'HOPFIELD' in the center. Below the title bar, there are three tabs: 'Train Data', 'Test Data', and 'Start Training'. The 'Train Data' tab is currently selected, showing a list of 15 binary vectors (e.g., 111111111, 1, 11111111, etc.). The 'Test Data' tab is also visible, showing a list of 15 binary vectors (e.g., 111111111, 1, 11111111, etc.). The 'Start Training' button is highlighted. Below the data lists, there is a 'Reset' button.

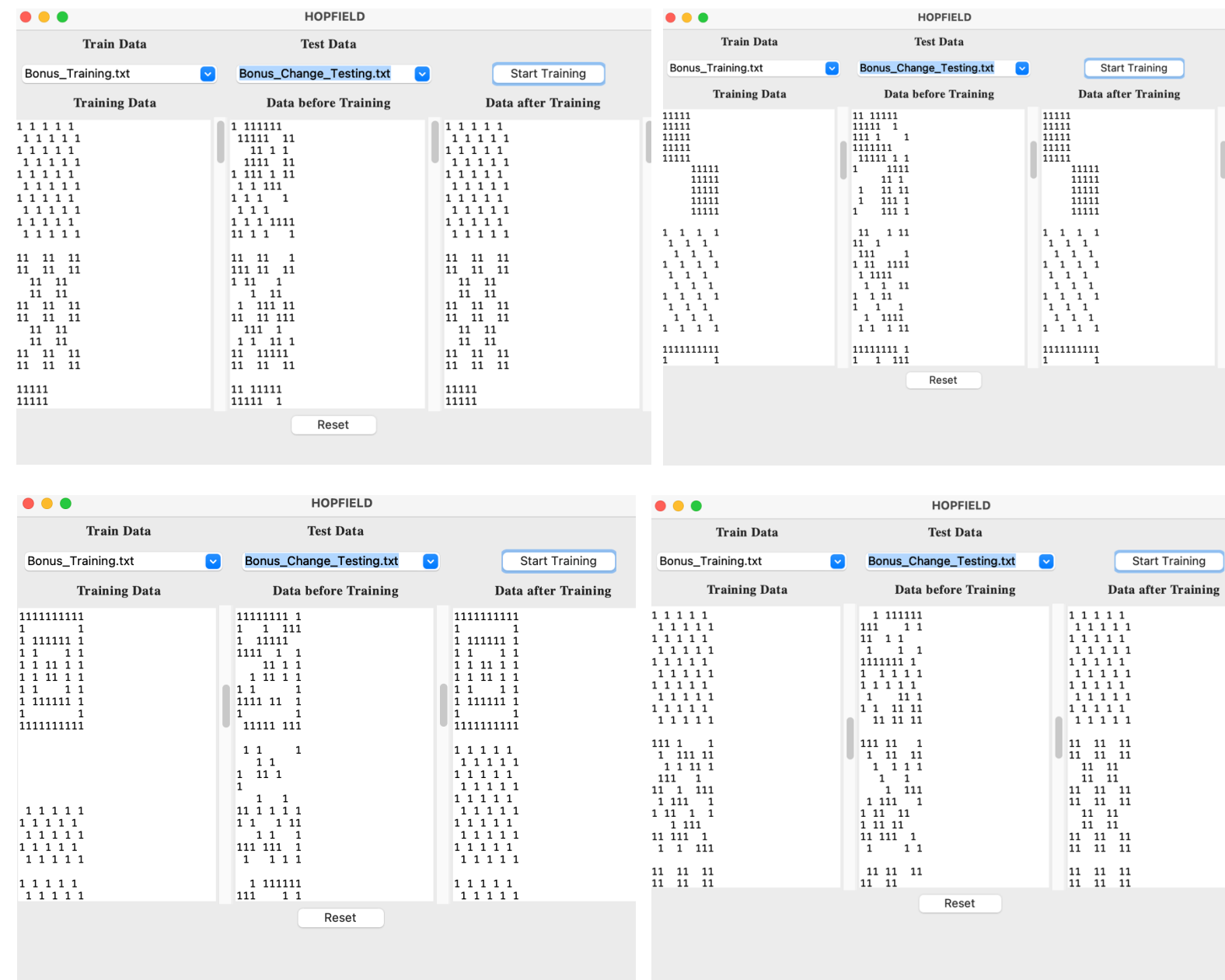
[illegible][illegible]

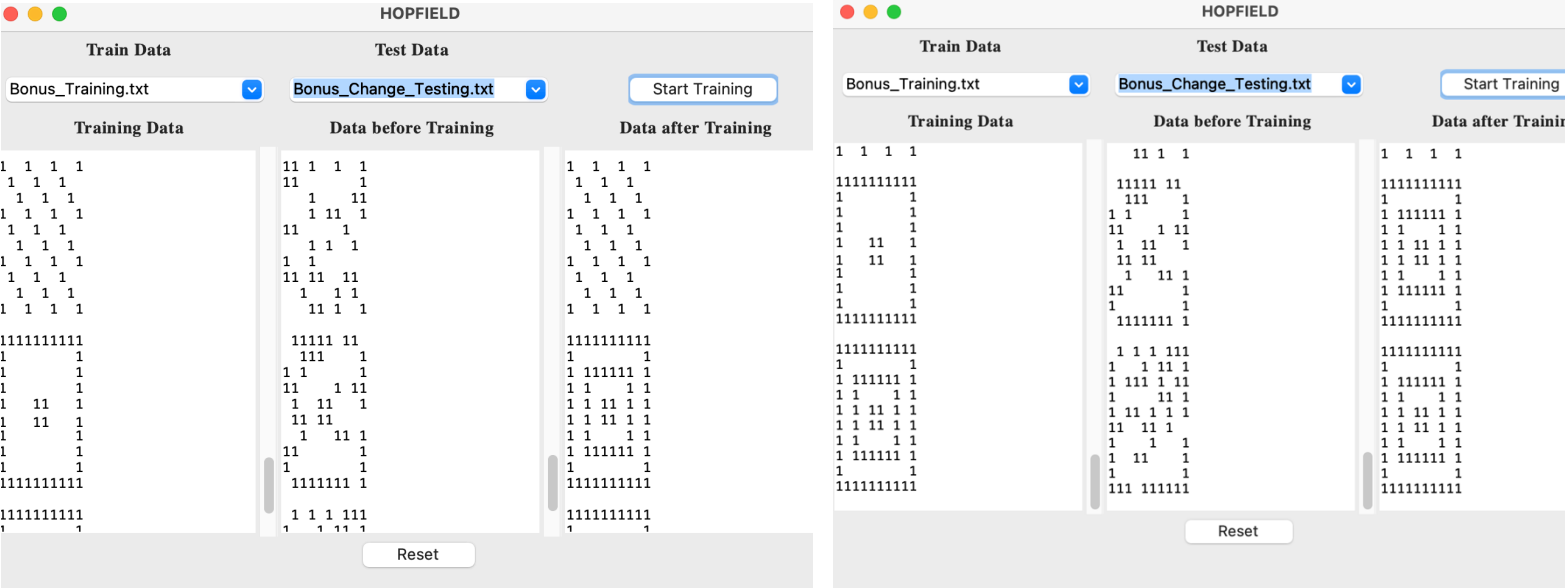
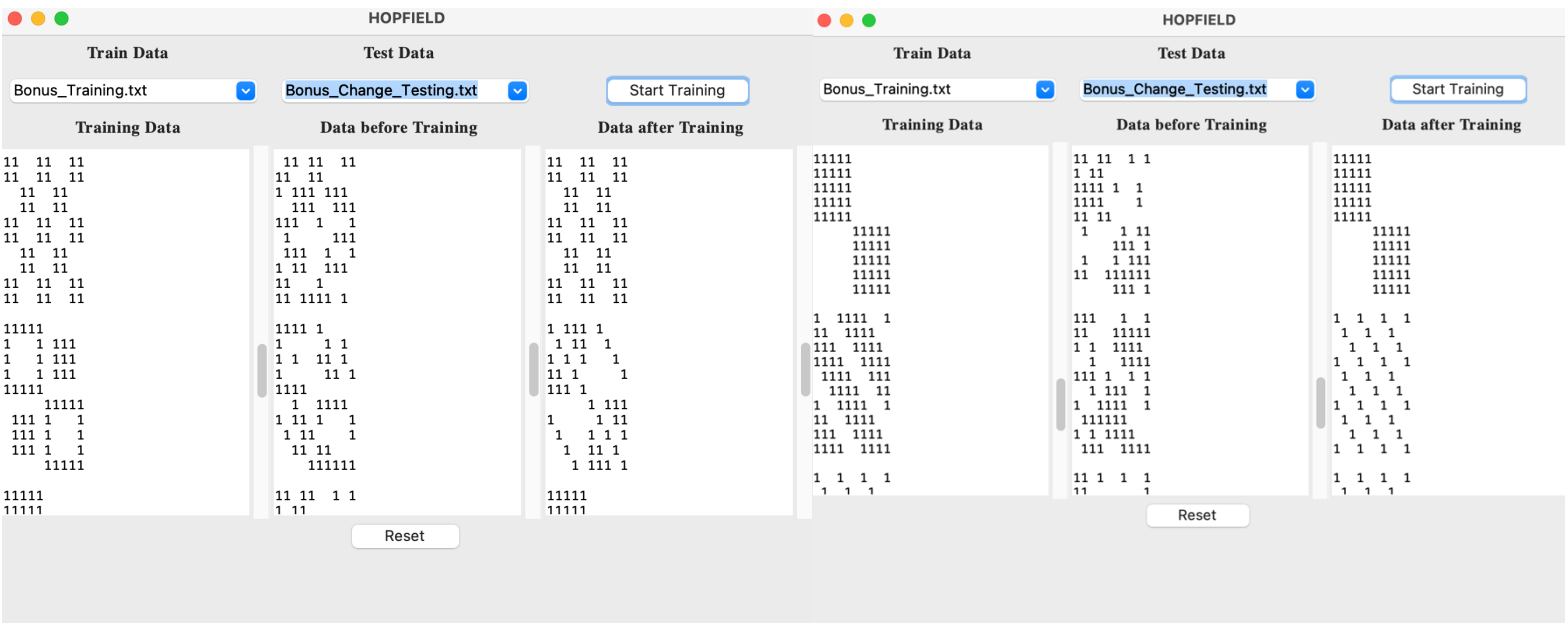
The screenshot shows a Mac OS-style window titled "HOPFIELD". At the top left are three colored window control buttons (red, yellow, green). The window is divided into three main vertical panels. The top of the window has two labels, "Train Data" and "Test Data", centered above their respective input fields. The "Train Data" input field contains "Bonus\_Training.txt" and has a blue dropdown arrow. The "Test Data" input field contains "Bonus\_Testing.txt" and also has a blue dropdown arrow. To the right of these input fields is a "Start Training" button with a blue border. Below the input fields, the three panels are labeled "Training Data", "Data before Training", and "Data after Training". Each panel contains a list of binary strings (0s and 1s). The "Training Data" panel has 15 lines of data. The "Data before Training" panel has 15 lines of data. The "Data after Training" panel has 15 lines of data. At the bottom center of the window is a "Reset" button.

Training Data	Data before Training	Data after Training
11111	11 11	11111
11111	1 1	11111
11111	1 1	11111
11111	11 11	11111
11111	11111	11111
11111	11111	11111
11111	11 1	11111
11111	1 1	11111
11111	1 11	11111
11111	11111	11111
11111	1 1111 1	1 1 1 1
11 1111 1	11 11 1	1 1 1
111 1111	1 1 1 1	1 1 1
1111 1111	1 11 11 1	1 1 1 1
1111 111	1111 1 1	1 1 1
1111 11	1 11 11	1 1 1
1 1111 1	1 1 11 1	1 1 1 1
11 1111	11 1 1	1 1 1
111 1111	1 1 1 1	1 1 1
1111 1111	1 11 1 1	1 1 1 1
1 1 1 1 1	1 1 1 1	1 1 1 1
1 1 1	1	1 1 1



(d) Bonus\_Change\_Testing.txt(加分)





#### 4. 實驗結果分析及討論

- 根據 3-實驗結果可知，基本題的測試資料(Basic\_Testing.txt)中所有圖樣經過一次迭代的非同步更新後，都可以回想正確。
- 根據 3-實驗結果可知，基本題的訓練資料加入雜訊後作為測試資料 (Basic\_Change\_Testing.txt) 中所有圖樣經過一次迭代的非同步更新後，都可以回想正確。
- 加分題的測試資料(Bonus\_Testing.txt)和加分題的訓練資料加入雜訊後作為測試資料(Bonus\_Change\_Testing.txt)：都只有部分圖樣能回想正確。推測應該是因為在訓練資料集中有些圖是有互相包含關係的。這樣會導致網路不知道要收斂誰。
- 基本題的圖樣都不互相包含，加分題的圖樣有些互相包含。所以我想這可以推論出，Hopfield 只能回想不互相包含的圖樣，對於記憶的圖樣有所限制。

## 5. 實作問題

一開始我將圖片中的空白轉為 0，1 轉為 1，但是在回想基本題的時候一直沒辦法回想成功。後來我將空白轉為-1，1 轉為 1 後，才能回想成功。

原本以為實作 Hopfield 的方法只有一種：藉由訓練資料得到鍵結值、利用鍵結值結合測試資料並和訓練資料做比較，再更新測試資料。但是在實作加分題時，發現加分題的訓練資料集中，有些圖是有互相包含關係的。這樣會導致網路不知道要收斂誰，沒辦法回想正確。但是我在網路上沒找到 Hopfield 可能的變形方法，所以加分題只有部分圖樣回想成功。

原本想自己做數字圖樣的訓練資料和測試資料，但是在執行程式的時候一直出錯。在 debug 的時候發現是因為資料裡有多的空格。但是我把空格都刪掉之後還是會出現一樣的錯。最後嘗試了很久都沒辦法成功，所以就只先回想助教提供的資料集。

一開始，我是用 wxPython 做 GUI 介面，但我發現在上傳圖樣到 GUI 上時，圖樣的格式會跑掉。所以我後來又再設計一個介面，但這個介面改用 tkinter 寫，這樣圖樣的格式就不會跑掉。而因為換了一個工具實作 GUI，所以當時也花了不少時間研究 tkinter 的使用方法。